

Figure 1: Pattern matcher test patterns for various applications.

Software Block Diagram

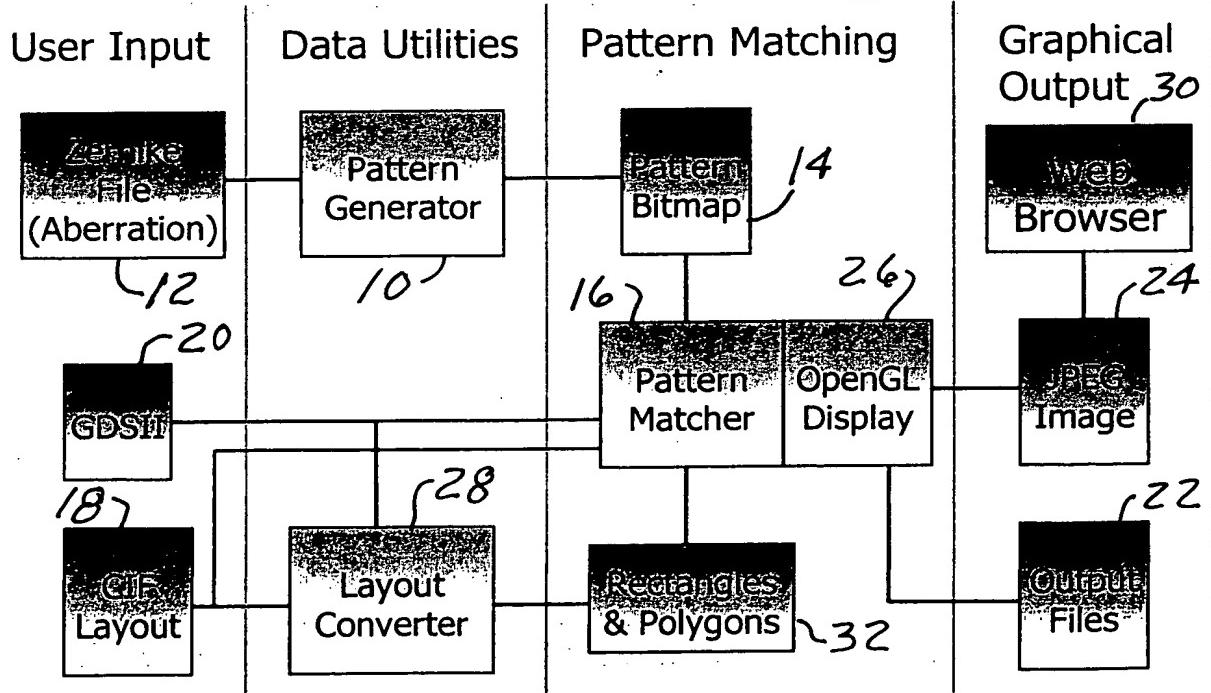
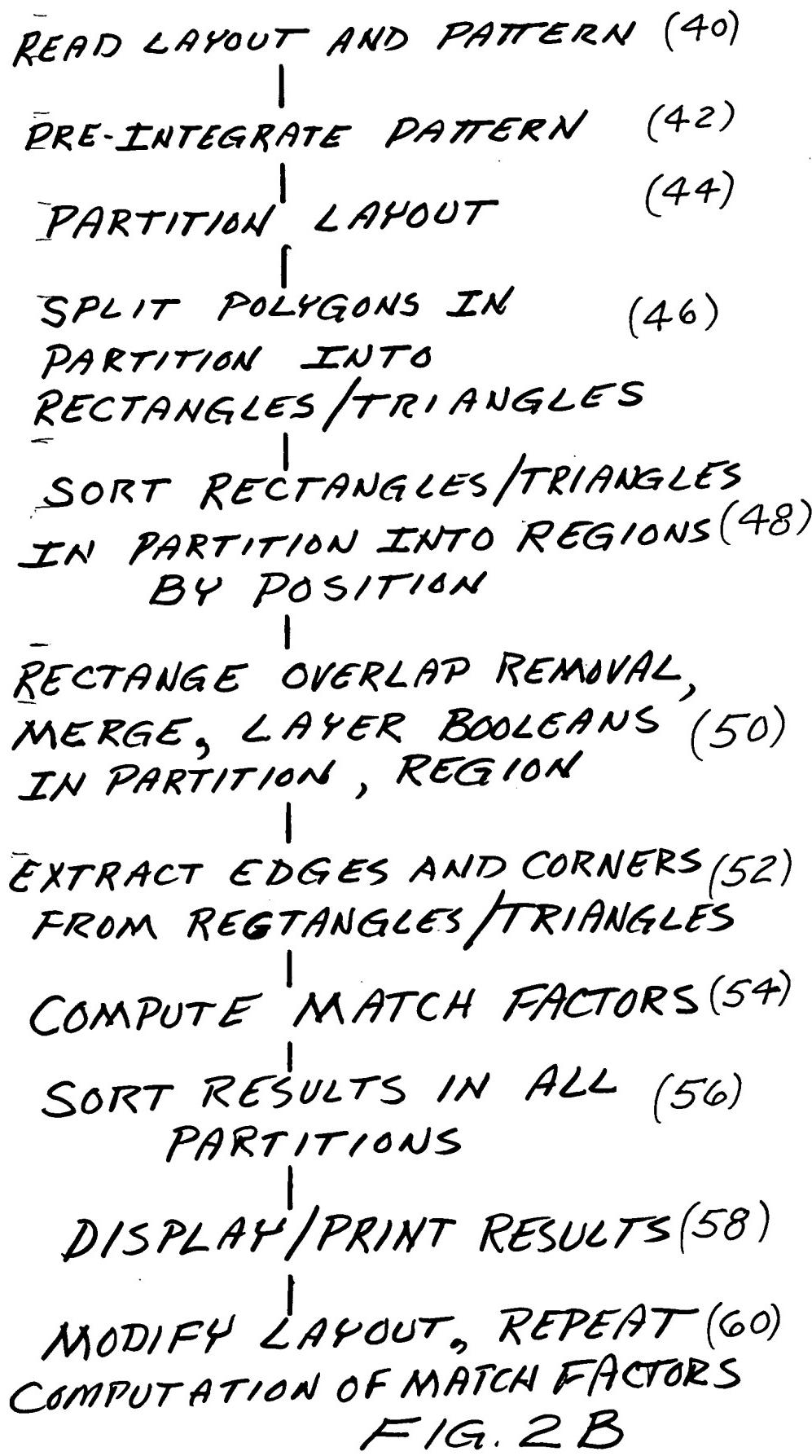


FIG. 2A



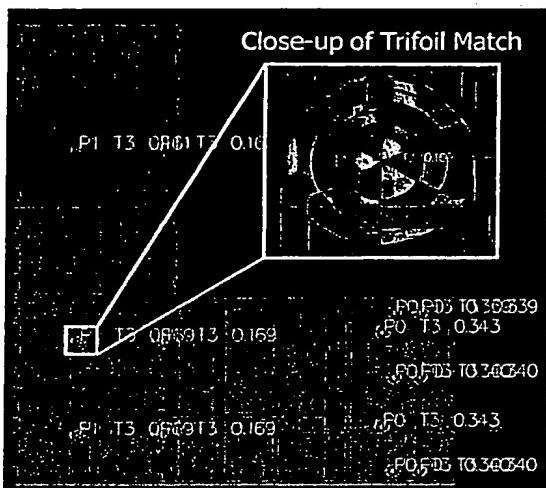


Figure 3: Trifoil and coma patterns matched on 0/180-degree FPGA interconnect layout.

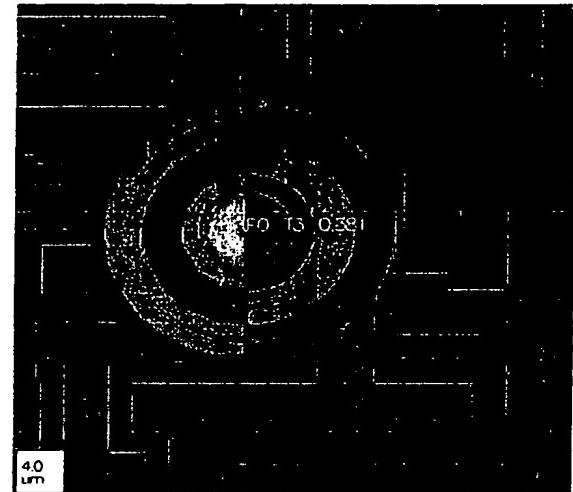


Figure 4: Coma pattern match on two-layer mask layout with 45-degree edges. The white square is 4 μ m.

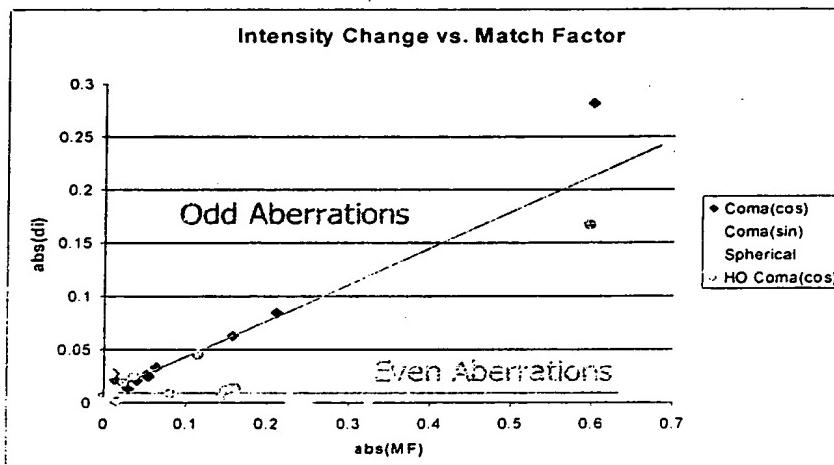


Figure 5: Simulated intensity change vs. match factor prediction for various aberration patterns and layouts.

Generic Pattern Matching Code

1. Divide input shapes (polygons) into geometric primitives
2. Spatially organize primitives by X, Y , etc.
3. Compute Match Factor (MF):

```
for each pattern P  
    for each orientation of P  
        for each match type T  
            for each  $X, Y$  match location  
                for each geom. Primitive G overlapping P  
                    add contribution of G on P at  $X, Y$  to MF
```

Time dominated by #3: #patterns x #orientations x #types
x #locations x #primitives_overlap_pattern x
time(primitive)

$\mathcal{F}/\mathcal{G}.$ 6

Data Structures

- Input = polygons, rectangles (special case of a polygon), paths (can be converted to polygons), and circles (can be approximated by many-sided polygons) = polygons
- Geometric Primitives:

Type	Number in layout	Operations to add to MF (time)
Pixel (Bitmap Alg.)	Very Large (area)	1
Edge Intersection	Large (perimeter)	2
Rectangle	Medium	4
Triangle	Small (or none)	4 to 12 (if split)

- Higher-level primitives (lower in table) are much more efficient to store and use

F/G. 7

Polygon Splitting (Bitmap)

- Manhattan Polygon => Bitmap
 - Too many pixels to store – large blocks of the same value

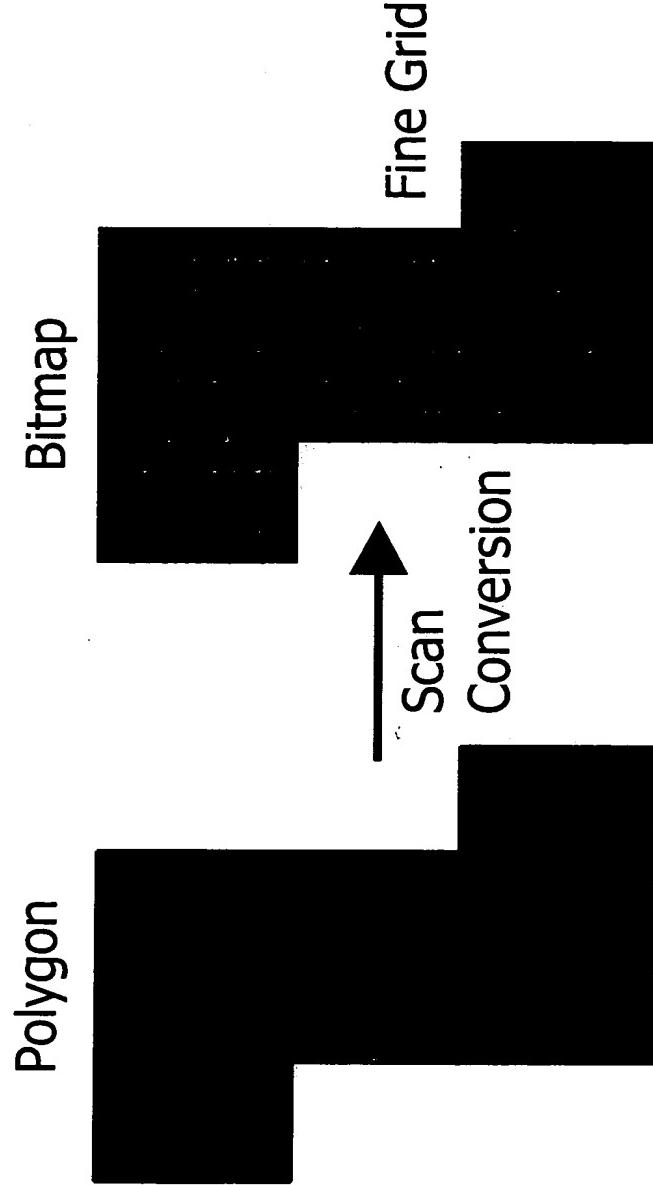
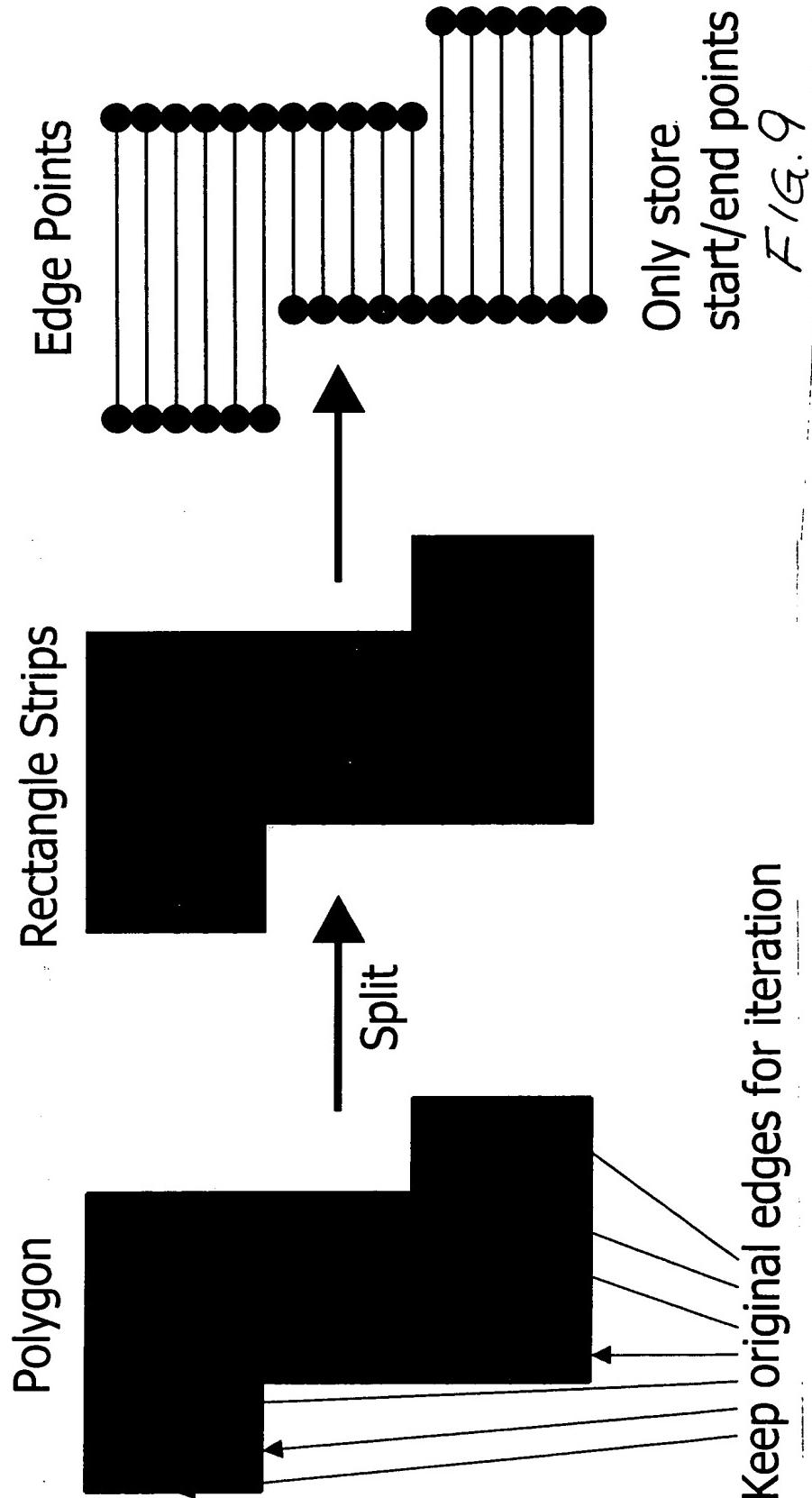


FIG. 8

Polygon Splitting (Edges)

- Manhattan Polygon => Edges
 - Well, actually rectangle strips between 2 edges



Polygon Splitting (Rectangles)

- Manhattan Polygon => Rectangles
- A
-
- Polygon → Split → Rectangles (final)
- The diagram shows a black L-shaped polygon at the bottom left. An arrow labeled "Split" points to a state where the polygon is divided into two separate L-shaped regions. A final arrow labeled "Rectangles (final)" points to a state where the polygon is composed of four solid black rectangles.
- Non-Manhattan Polygon => Rectangles
- B
-
- Polygon → Split → (Lots of) Rectangles
- The diagram shows a black polygon with a diagonal edge at the bottom left. An arrow labeled "Split" points to a state where the polygon is divided into many small, thin black rectangles, described as "Lots of Rectangles".
- C
-
- Polygon → Split → H, V Merge → Still Lots of Rectangles
- The diagram shows a black polygon with a diagonal edge at the bottom left. An arrow labeled "Split" points to a state where the polygon is divided into many small, thin black rectangles. A second arrow labeled "H, V Merge" points to a state where the rectangles are merged into larger, thicker black rectangles, resulting in "Still Lots of Rectangles".
- Non-Manhattan Polygon => Rectangles
- D
-
- Polygon → Split → Steps on fine grid
- The diagram shows a black polygon with a diagonal edge at the bottom left. An arrow labeled "Split" points to a state where the polygon is divided into many small, thin black rectangles. A second arrow labeled "Steps on fine grid" points to a state where the rectangles are merged into larger, thicker black rectangles, resulting in "Steps on fine grid".

FIG. 10

Polygon Splitting (Triangles)

- Non-Manhattan Polygon => Rectangles + Right Triangles

Primary Goal: Min # Triangles
Secondary Goal: Min # Rectangles

Polygon
Rectangles and Right
45 degree Triangles

Rectangles and Right
45 degree Triangles

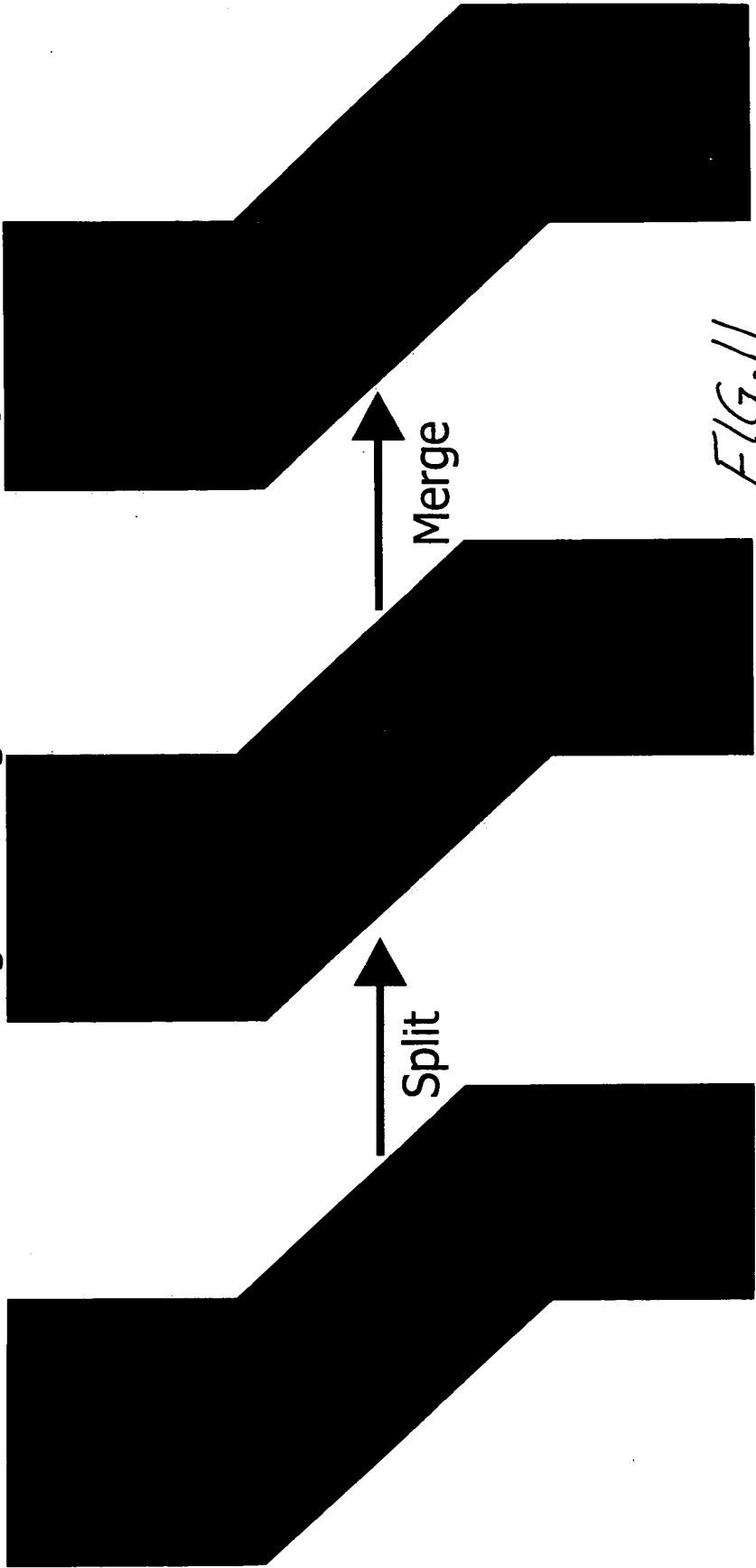


FIG.11

Pattern Pre-Integration

- 1D Pre-Integration
 - Can be horizontal or vertical, either will work
 - Pre-integrated value = sum of all pattern values at and to the right

Pattern values	0	1	2	1	3	0	1
Pre-int values	0	8	8	7	5	4	1

Typical PM pattern is 128x128

- 2D Pre-Integration
 - Starts with 1D pre-integration
 - Pre-integrated value = sum of all pattern values at and to the right AND above (top right = orientation P0)

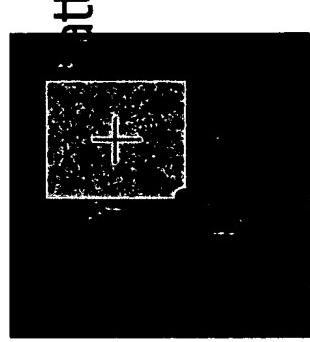
Pattern Values	0	1	2	1	2	1	2	1	0	0	0	PV
IR (IU)	4	4	3	3	2	2	2	2	1	1	1	above

F/G.	12
------	----

Algorithm 1: Bitmap

- Entire layout represented as one huge bitmap of layers (like images on a computer screen)
- One rectangle is added at a time to the bitmap
- At every match location (edge, corner, etc.), each pattern pixel is multiplied by the layout pixel and summed:

$$MF(i + \frac{X}{2}, j + \frac{Y}{2}) = norm * \sum_Y \sum_X Layout(x+i, y+j) * Pat(x, y)$$



- Pattern size (X by Y) is typically 128x128
- = 16384 ops

FIG. 13

Algorithm 2: Edge Intersections

- Store only the pixels along edges
- Run-length encoding in 1D – skip large runs of the same pixel value (rectangle strips)
- Pre-integrate pattern in 1D: $val(i, j) = \sum_{k=i}^X pat(k, j)$ for x intersection case
- Add MF contributions from each rectangle strip between two edges (either X or Y dir)

pat(...,j)	0	1	2	1	3	0	1
val(...,j)	8	8	7	5	4	1	1
r strip (weight 1)	1						-1

+ edges -

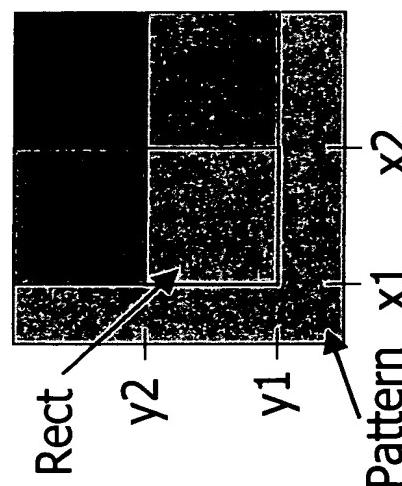
Contribution:

$$1*8 + (-1)*1 = 7$$

$F/G_i / 4$

Algorithm 3: Rectangles

- Simplest data structure: Store only the rectangles and pointers to them
- 2D encoding – only rectangle corners are needed
- Pattern integrated in 2D, rectangle LL corner clipped to pattern area
- Integrated pattern value is sum of values above and to the right: $val(i, j) = \sum_{k=i}^Y \sum_{l=j}^X pat(k, l)$



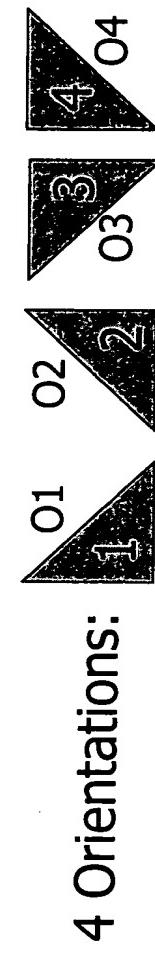
Contribution from rect at $(x_1, y_1), (x_2, y_2) =$
 $val(x_1, y_1) - val(x_2, y_1) - val(x_1, y_2) + val(x_2, y_2)$

$\mathcal{F}/G / 5$

Only process LL corner and other 3 if inside pattern

Algorithm 3b: Triangles

- Extension of rectangle algorithm
- Pre-integration time/storage proportional to the number of unique angles
 - Limited to multiples of 45-degree angles in practice
 - 0, 45, 90, 135, 180, 225, 270, 315 deg => 8 pre-integrations



Triangle clipping
is difficult

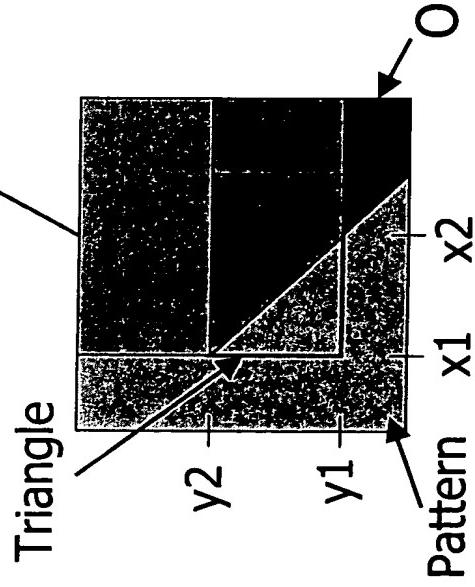
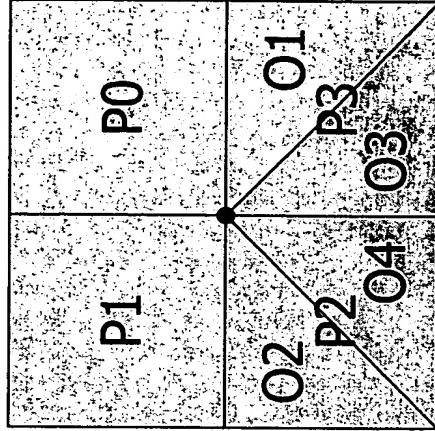
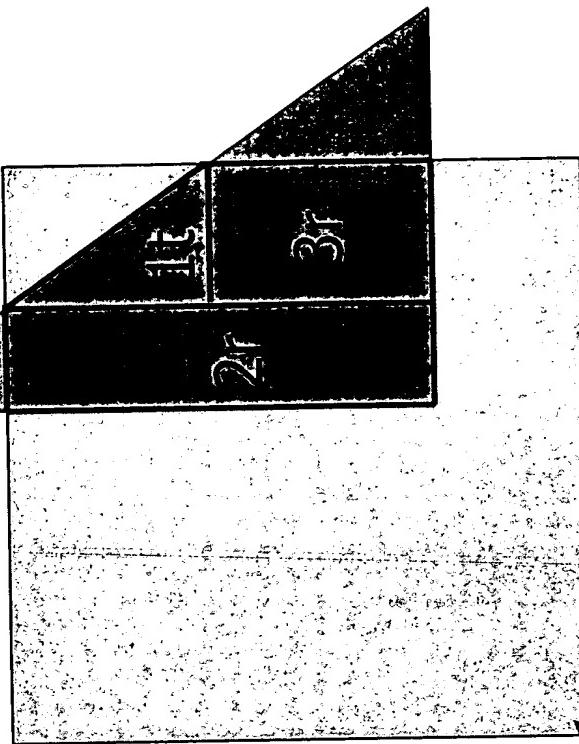
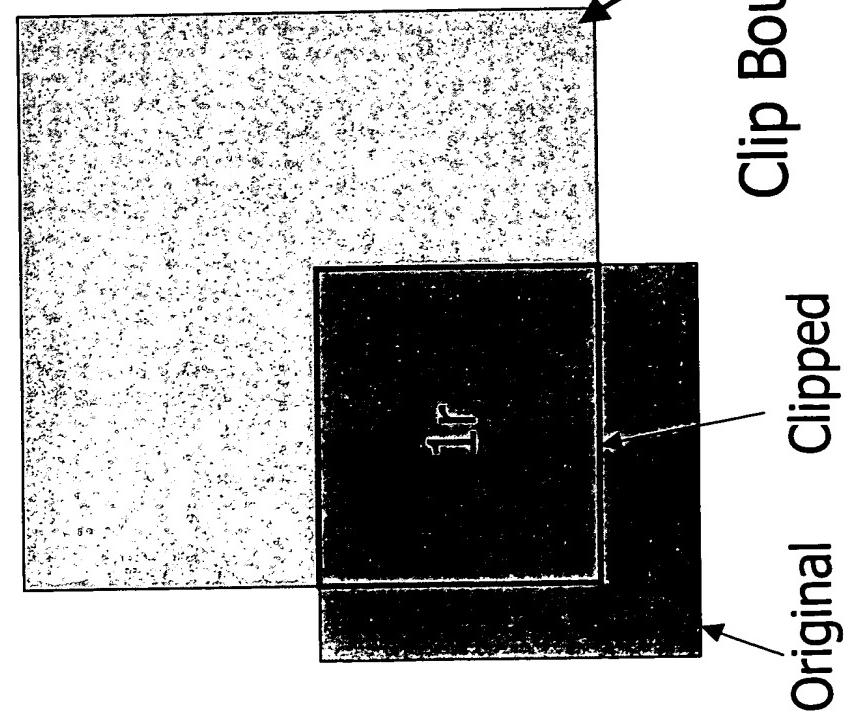


FIG. 16

Rectangle/Triangle Clipping

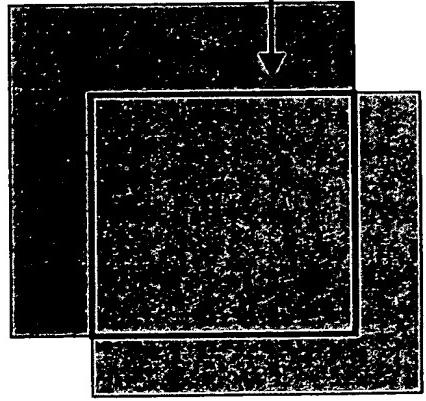
A Rectangle => Rectangle
B Triangle => Triangle
(+ Rectangles)



Original Clipped
Clip Boundary = Pattern Area
Clipped (up to 3 pieces)
F/G. 17

Pattern

Examples



Input Rectangle

RL = rectangle length (3)
RH = rectangle height (3)
TL = triangle length (3)
TH = triangle height (3)

Bitmap Algorithm

Pattern Values

PV	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
IR	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	

Bitmap Algorithm

Pattern Values

PV	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
IR	1	2	3	4	5	6	7	8																																																																																													

Examples

Rectangle Algorithm

45-Triangle Algorithm

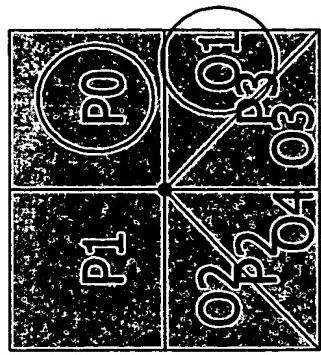
2D Pre-Int top right

			P0
1	4	3	1
10	7	6	3
18	11	9	5

8-way Pre-Int → Precomputed:

P0 from rect algorithm

		PV
Pattern	1	2
Values	3	4
0B	1	2
3	1	2
4	1	2
2A	2	0
		0



$$\text{LLC} - \text{ULC} - \text{LPC} + \text{URC} = \\ 22 - 4 - 5 + 1 = 14 \\ \text{Always 4 Operations}$$

$$\text{P0(A)} - \text{P0(B)} - \text{O1(B)} + \text{O1(C)} = \\ 11 - 4 - 5.5 + 0 = 1.5 \\ \textbf{4 Operations/Shape (12 max)}$$

F/G. /9

Examples

1D Pre-Int to the right

	IR		
4	3	1	2
6	3	3	2
8	4	3	2
4	2	0	0

2D Pre-Int top right

	P0		
4	3	1	3
10	7	6	3
18	11	9	5
22	13	9	5

Non-45 degree Triangle (Proposed)

B	PV	P0(A) - P0(B) - IR(B...C) =
Pattern	3 0 1 2	$18 - 0 - (4 + 3 + 3) = 8$
Values	4 1 1 2	$TH + 2 = 5$ Operations

$$\begin{aligned}
 & P0(A) - P0(B) - IR(B...C) = \\
 & 18 - 0 - (4 + 3 + 3) = 8 \\
 & TH + 2 = 5 \text{ Operations}
 \end{aligned}$$

Similar to edge intersection
algorithm but reduced storage
F/G. 20

Data Structures and Algorithms

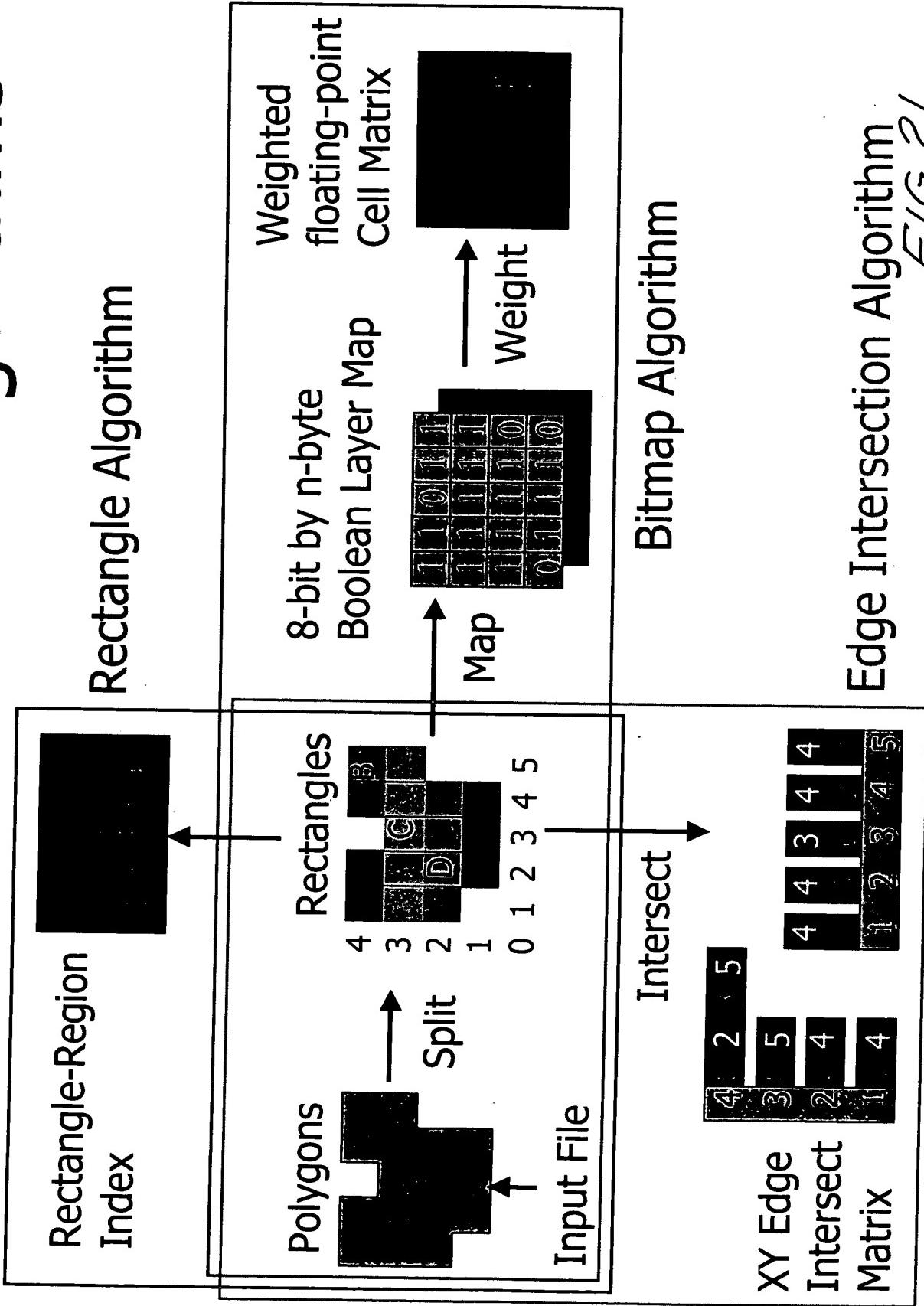


FIG. 21